

Security Assessment of the Padak Village Administration and Letter Service Information System Using STRIDE, OWASP ZAP, OWASP Top 10 (2021), and the OWASP Risk Rating Methodology

Sophia¹, Yuliana², Santi Thomas³, and Devon Surya Pranata⁴

¹²³⁴*Information Technology, Institut Shanti Bhuana, Bengkayang, Kalimantan Barat Indonesia*

**Corresponding author: sophia@shantibhuana.ac.id*

ABSTRACT

The Padak Village Administration and Letter Service Information System is a web-based application that supports administrative services for citizens, making security evaluation essential to identify threats, vulnerabilities, and risk levels that may affect system security. This study aims to identify security threats, discover vulnerabilities, determine risk levels, and propose mitigation recommendations to improve system security. A descriptive approach was employed by integrating the STRIDE method for threat identification, OWASP ZAP (AJAX Spider and Active Scan) for vulnerability assessment, OWASP Top 10 (2021) for vulnerability classification, and the OWASP Risk Rating Methodology for risk assessment based on Likelihood and Impact factors. OWASP ZAP identified 28 security alerts during the scanning process. However, only 21 vulnerabilities were included in the risk assessment because the remaining 7 findings were classified as Informational, which only describe application configuration and characteristics without representing directly exploitable security vulnerabilities. The STRIDE analysis identified four categories of security threats, namely Spoofing, Tampering, Repudiation, and Information Disclosure. Furthermore, the 21 selected vulnerabilities were mapped into six OWASP Top 10 (2021) categories: A01:2021 Broken Access Control, A02:2021 Cryptographic Failures, A03:2021 Injection, A05:2021 Security Misconfiguration, A07:2021 Identification and Authentication Failures, and A08:2021 Software and Data Integrity Failures. Based on the OWASP Risk Rating Methodology, the assessment identified 2 Critical, 15 High, 3 Medium, and 1 Low-risk vulnerabilities. The study proposes mitigation recommendations, including the implementation of Prepared Statements, Anti-CSRF Tokens, Content Security Policy (CSP), HTTP Security Headers, HTTPS, Subresource Integrity (SRI), strengthened Cross-Origin Resource Sharing (CORS) configuration, secure Cookie attributes, and application security hardening in accordance with OWASP ZAP recommendations.

Keywords: OWASP Top 10, OWASP ZAP, Risk Assessment, STRIDE, Web Application Security

1. INTRODUCTION

The rapid development of information technology has significantly influenced various sectors, including village government administration. The implementation of web-based information systems enables administrative processes to become faster, more efficient, and better organized. The Padak Village Administration and Letter Service Information System is used to manage various public services, including the issuance of business certificates, domicile certificates, certificate ownership statements, death certificates, and certificates of indigence.

The system stores various types of critical data, including citizens' personal information, administrative records, and document archives used in public service processes. Therefore, the Padak Village Administration and Letter Service Information System constitutes an important

information asset that must be protected from potential security threats. If the system does not provide an adequate level of security, risks such as data leakage, misuse of citizens' personal information, and manipulation of administrative records may occur, potentially causing harm to both the village government and the community. Since the system processes sensitive citizen and administrative data, conducting a security assessment is not only a technical requirement but also an ethical responsibility to ensure the confidentiality, integrity, and availability of public service information. Furthermore, the system utilizes Single Sign-On (SSO) authentication through Google accounts to simplify the user login process. Single Sign-On technology enables users to access the system using a single authentication account without the need to create separate credentials. The implementation of SSO is widely adopted in web-based applications because it enhances user convenience and improves authentication efficiency (Mukrodin, Taufiq, and Rayananda 2023). Despite these advantages, the use of third-party authentication services such as Google SSO may also introduce security risks if not implemented correctly. Research on Single Sign-On implementation has shown that configuration or implementation errors can create opportunities for attacks, including authentication token misuse and unauthorized access to information systems (Salmuasih and Setiawan 2023). Therefore, information systems utilizing modern authentication technologies such as Google SSO should undergo security assessment to ensure that the authentication mechanism does not introduce new security vulnerabilities. The security assessment was conducted on the Padak Village Administration and Letter Service Information System, which was developed by the researcher during an internship program at the Padak Village Office based on functional requirements and recommendations provided by the village staff. The system was developed as a prototype to support village administrative services, while functional testing, including the submission of administrative letter requests, was performed by the researcher using representative test data in a controlled development environment prior to operational deployment. Formal authorization to use the system as the object of this research was obtained from the Padak Village Office. The security assessment was conducted within the scope of this authorized academic research to identify security vulnerabilities and formulate mitigation recommendations without processing or exposing actual citizen data. Threat modeling is a fundamental activity in secure software development because it enables security threats to be systematically identified before exploitation. One of the most widely adopted threat modeling approaches is STRIDE, developed by Microsoft, which classifies threats into Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege (Microsoft 2024). To validate identified threats, automated vulnerability assessment tools such as OWASP ZAP can be employed, while (Foundation 2021) published by the OWASP Foundation provides an internationally recognized framework for classifying web application security risks.

Recent studies have employed different approaches to improve web application security. (Das et al. 2024) applied the STRIDE methodology to identify cybersecurity threats and assessed them using the SAHARA and DREAD methods. Although their study demonstrated the effectiveness of STRIDE for systematic threat modeling, it did not validate the identified threats through automated vulnerability scanning or classify the findings using the OWASP Top 10 and the OWASP Risk Rating Methodology. Similarly, (Rifat, Sultana, and Hossain 2023) utilized OWASP ZAP to assess vulnerabilities in government and financial websites and

proposed mitigation strategies; however, the study focused on vulnerability detection without integrating STRIDE-based threat modeling or systematic risk prioritization. Furthermore, (Wisnu and Soewito 2026) evaluated application security using the OWASP Top 10 framework together with SonarQube and OWASP ZAP, but their study employed CVSS for severity assessment and did not establish traceability between threat identification, vulnerability validation, and risk assessment. Therefore, previous studies have generally applied STRIDE, automated vulnerability scanning, or the OWASP Top 10 independently or only partially integrated these approaches. The linkage between threat modeling, vulnerability validation, vulnerability classification, formal risk assessment, and mitigation remains limited. To address this gap, this study integrates STRIDE, OWASP ZAP Active Scan, OWASP Top 10 (2021), and the OWASP Risk Rating Methodology into a unified security assessment workflow. This integration enables systematic traceability from DFD-based threat identification, vulnerability validation, vulnerability classification, risk prioritization, and mitigation recommendations for the Padak Village Administration and Letter Service Information System.

Unlike previous studies that primarily applied STRIDE, automated vulnerability scanning, or the OWASP Top 10 independently, this study integrates these approaches into a unified security assessment workflow. The proposed workflow establishes traceability from Data Flow Diagram (DFD)-based threat identification using STRIDE, vulnerability validation through OWASP ZAP Active Scan, vulnerability classification according to the OWASP Top 10 (2021), risk evaluation using the OWASP Risk Rating Methodology, and mitigation prioritization based on the assessed risk level. By connecting these complementary security assessment activities, the proposed approach provides a more systematic and transparent process for evaluating web application security than applying each method separately. The main contributions of this study are threefold. First, it proposes an integrated security assessment workflow that combines STRIDE, OWASP ZAP Active Scan, OWASP Top 10 (2021), and the OWASP Risk Rating Methodology into a single assessment process. Second, it establishes traceability between threat identification, vulnerability validation, vulnerability classification, risk assessment, and mitigation recommendations, enabling each mitigation decision to be supported by identified threats and validated technical evidence. Third, it provides a prioritized set of mitigation recommendations for the Padak Village Administration and Letter Service Information System based on formal risk assessment, which can serve as a practical reference for improving the security of similar web-based public service applications.

2. METHODS

This study evaluates the security of the Padak Village Administration and Letter Service Information System by integrating STRIDE Threat Modeling, OWASP ZAP Active Scan, OWASP Top 10 (2021), and the OWASP Risk Rating Methodology. The assessment was conducted on a development version of the system developed by the researcher during an internship program at the Padak Village Office based on functional requirements and recommendations provided by the village staff. Formal authorization to use the system as the object of this research was obtained from the Padak Village Office. Functional testing and security assessment were performed in a controlled development environment using representative test data without processing actual citizen information. Primary data were collected through direct observation of the implemented system, analysis of system

documentation and functional requirements, review of the application architecture, and inspection of the system workflows used to construct the Data Flow Diagrams (DFDs). The DFD Level 0 and Level 1 models were developed based on the implemented system architecture and verified by comparing the diagrams with the application's functional modules, user interactions, and database processes to ensure consistency with the actual system. The security assessment considered two user roles, namely administrator and user, representing the access privileges implemented in the application. The research methodology consisted of six stages. First, DFD Level 0 and Level 1 were constructed to represent the system architecture. Second, STRIDE Threat Modeling was applied to identify potential threats associated with each DFD element. Third, OWASP ZAP Active Scan was employed to identify technical vulnerabilities, while Passive Scan was used to obtain additional security findings. Fourth, the confirmed vulnerabilities were classified according to the OWASP Top 10 (2021). Fifth, the OWASP Risk Rating Methodology was applied to evaluate the likelihood, impact, and overall risk level of each confirmed vulnerability. Finally, mitigation recommendations were prioritized based on the assessed risk levels to improve the security of the system. The overall research methodology is illustrated in Figure 1.

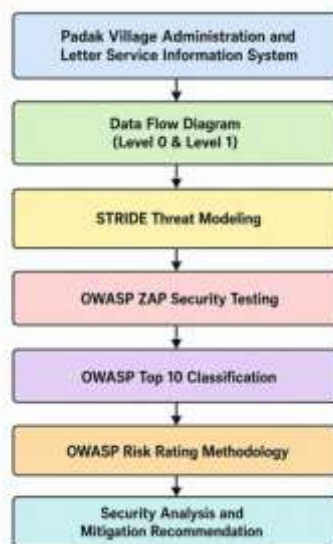


Figure 1. Security Analysis Methodology

Figure 1 illustrates the research workflow adopted in this study. The process begins with DFD modeling, followed by STRIDE threat identification, OWASP ZAP vulnerability scanning, OWASP Top 10 classification, OWASP Risk Rating assessment, and mitigation recommendation. Each stage builds upon the results of the previous stage to provide a comprehensive security evaluation of the system.

2.1 STRIDE Threat Modeling

STRIDE Threat Modeling was employed to identify potential security threats associated with the system architecture represented by the Data Flow Diagrams (DFDs). Each process, data flow, external entity, and data store was analyzed according to the six STRIDE threat categories: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and

Elevation of Privilege. Since STRIDE identifies potential threats at the architectural level, the identified threats were subsequently correlated with the results of the OWASP ZAP Active Scan to determine whether observable web application vulnerabilities provided technical evidence supporting the identified threats. Because OWASP ZAP is designed to identify technical vulnerabilities rather than validate all architectural threats, only STRIDE threats associated with detectable web application weaknesses were supported by OWASP ZAP findings. The traceability between the identified STRIDE threats and the corresponding OWASP ZAP evidence used in this study is presented in Table 1.

Table 1. Traceability of STRIDE Threats Supported by OWASP ZAP Findings

DFD Element	STRIDE Threat	Example Case	OWASP ZAP Evidence
Google SSO Login	Spoofing	Authentication requests may be forged due to the absence of request validation mechanisms.	Absence of Anti-CSRF Tokens, Cross-Domain Misconfiguration, Missing Anti-clickjacking Header
Google SSO Login	Tampering	Authentication requests may be modified during transmission.	Absence of Anti-CSRF Tokens
Google SSO Login	Information Disclosure	Authentication information may be exposed because of insecure communication and security header misconfigurations.	Cross-Domain Misconfiguration, HTTPS Content Available via HTTP, Strict-Transport-Security Header Not Set
Letter Submission Process	Tampering	Malicious input may modify or manipulate application data through user input forms.	SQL Injection, Absence of Anti-CSRF Tokens
Letter Verification Process	Tampering	Administrative requests may be modified through forged requests.	Absence of Anti-CSRF Tokens
Login Data Flow	Information Disclosure	Authentication data may be	HTTPS Content Available via

DFD Element	STRIDE Threat	Example Case	OWASP ZAP Evidence
		exposed because secure communication policies are not fully enforced.	HTTP, Strict-Transport-Security Header Not Set, Content Security Policy (CSP) Header Not Set
User and Letter Database	Tampering	Database contents may be modified through SQL Injection attacks.	SQL Injection
User and Letter Database	Information Disclosure	Sensitive information stored in the database may be exposed through security misconfigurations.	SQL Injection, Directory Browsing, Content Security Policy (CSP) Header Not Set

As shown in Table 1, only STRIDE threats that could be associated with observable web application vulnerabilities were supported by OWASP ZAP findings. The confirmed vulnerabilities were subsequently mapped to the OWASP Top 10 (2021) categories to determine the corresponding security risk classification. Each confirmed vulnerability was then evaluated using the OWASP Risk Rating Methodology to determine its likelihood, impact, and overall risk level. Finally, mitigation recommendations were prioritized according to the assessed risk level. This sequential process establishes traceability from architectural threat identification to vulnerability validation, vulnerability classification, formal risk assessment, and mitigation planning.

2.2 Vulnerability Testing Using OWASP ZAP

Following the identification of potential security threats using the STRIDE threat modeling approach, vulnerability testing was conducted using OWASP ZAP (Zed Attack Proxy) version 2.17.0. OWASP ZAP is an open-source web application security testing tool that identifies security vulnerabilities by analyzing HTTP and HTTPS communications between a client and a web application through an interception proxy. In this study, OWASP ZAP was used for automated vulnerability scanning rather than penetration testing because the objective was to identify security weaknesses without exploiting the detected vulnerabilities. The security assessment was performed by configuring Mozilla Firefox to communicate with OWASP ZAP through a local proxy, enabling all HTTP and HTTPS requests and responses exchanged between the browser and the Padak Village Administration and Letter Service Information System to be intercepted and recorded. The application was manually explored using both User and Administrator accounts to ensure that all accessible features and endpoints were captured. The explored functionalities included Google Single Sign-On (Google SSO) authentication, letter submission, document download, signed document upload, data modification, and record deletion. Consequently, all recorded URLs, HTTP requests, HTTP responses, parameters, and

application endpoints were included within the scope of the security assessment. After the manual exploration phase, the AJAX Spider feature of OWASP ZAP was executed to crawl the application and discover dynamically generated pages and JavaScript-based content. The crawling process produced a complete application site tree containing the identified URLs, endpoints, and parameters. This site tree was subsequently used as the target for the vulnerability assessment. The security assessment was limited to application functionalities that were accessible through the available User and Administrator accounts. No custom scan rules were enabled, and the Active Scan was performed using the default OWASP ZAP scan policy with a Medium Alert Threshold and Medium Attack Strength. Since the objective of this study was vulnerability assessment rather than penetration testing, no exploitation of identified vulnerabilities was performed. Instead, the assessment focused on identifying observable security weaknesses within the authorized scope of the application. An Active Scan was then performed using the default scan policy with Medium Alert Threshold and Medium Attack Strength, while passive scanning was automatically conducted during user interaction with the application. The vulnerability assessment was generated on 9 July 2026 using OWASP ZAP version 2.17.0. The scan results were automatically classified into four risk levels: High, Medium, Low, and Informational. A total of 28 security findings were identified, consisting of 1 High, 11 Medium, 9 Low, and 7 Informational alerts. Only the 21 non-informational findings were included in the subsequent vulnerability analysis because informational alerts describe application characteristics or configuration information rather than exploitable security vulnerabilities. To improve the reliability of the assessment results, each non-informational finding reported by OWASP ZAP was manually reviewed by examining the corresponding HTTP requests, HTTP responses, affected URLs, and supporting evidence generated by the tool. This review was performed to ensure that the reported findings were relevant to the evaluated application before they were classified according to the OWASP Top 10 (2021) and assessed using the OWASP Risk Rating Methodology. The validated vulnerabilities were subsequently classified according to the OWASP Top 10 (2021) to determine their corresponding security categories. The classified vulnerabilities were then evaluated using the OWASP Risk Rating Methodology to determine their likelihood, technical impact, business impact, and overall risk level. These risk levels subsequently served as the basis for prioritizing mitigation recommendations.

2.3 Vulnerability Classification Using OWASP Top 10

The vulnerabilities identified from OWASP ZAP scanning were classified based on the OWASP Top 10 (2021) framework. OWASP Top 10 is an internationally recognized standard for identifying the most critical security risks in web applications. The OWASP Top 10 framework consists of ten vulnerability categories: Broken Access Control, Cryptographic Failures, Injection, Insecure Design, Security Misconfiguration, Vulnerable and Outdated Components, Identification and Authentication Failures, Software and Data Integrity Failures, Security Logging and Monitoring Failures, Server-Side Request Forgery (SSRF). Each validated vulnerability identified through OWASP ZAP was mapped to the most appropriate OWASP Top 10 (2021) category based on the official OWASP classification guidelines. This mapping established the relationship between the observed technical vulnerabilities and standardized web application security risks, thereby providing a consistent basis for the subsequent OWASP Risk Rating assessment. Following the vulnerability classification

process, each confirmed vulnerability was evaluated using the OWASP Risk Rating Methodology to determine its likelihood, technical impact, business impact, and overall risk level. The resulting risk levels were subsequently used to prioritize mitigation recommendations.

2.4 Risk Assessment

The final stage of the security assessment was risk evaluation using the OWASP Risk Rating Methodology. This methodology was applied to determine the severity level of each confirmed vulnerability by evaluating both the likelihood of successful exploitation and its potential technical and business impacts. The OWASP Risk Rating Methodology adopts a structured judgment approach, in which each assessment factor is evaluated using predefined scoring criteria provided in the official OWASP Risk Rating guidelines. Accordingly, the researcher, acting as the system assessor, assigned scores by selecting the most appropriate predefined values for each subfactor based on the characteristics of the confirmed vulnerabilities identified through the OWASP ZAP Active Scan. This approach provides a systematic and consistent basis for evaluating security risks while reducing subjectivity in the assessment process. The risk assessment consists of three sequential stages: likelihood calculation, impact calculation, and overall risk determination.

1. Likelihood Calculation

Likelihood represents the probability that a vulnerability can be successfully exploited. It is calculated by combining two factors: Threat Agent (Skill Level, Motive, Opportunity, and Size) and Vulnerability (Ease of Discovery, Ease of Exploit, Awareness, and Intrusion Detection).

$$\text{Threat Agent} = \frac{(\text{Skill Level} + \text{Motive} + \text{Opportunity} + \text{Size})}{4}$$

$$\text{Vulnerability} = \frac{(\text{Ease of Discovery} + \text{Ease of Exploit} + \text{Awareness} + \text{Intrusion Detection})}{4}$$

The final Likelihood value is calculated by combining the Threat Agent and Vulnerability values:

$$\text{Likelihood} = \frac{\text{Threat Agent} + \text{Vulnerability}}{2}$$

2. Impact Calculation

Impact measures the consequences of successful exploitation by evaluating Technical Impact (Confidentiality, Integrity, Availability, and Accountability) and Business Impact (Financial Damage, Reputation Damage, Non-Compliance, and Privacy Violation).

$$\text{Technical Impact} = \frac{(\text{Confidentiality} + \text{Integrity} + \text{Availability} + \text{Accountability})}{4}$$

$$\text{Business Impact} = \frac{(\text{Financial Damage} + \text{Reputation} + \text{Non Compliance} + \text{Privacy Violation})}{4}$$

$$\text{Impact} = \frac{\text{Technical Impact} + \text{Business Impact}}{2}$$

3. Risk Calculation

After obtaining the Likelihood and Impact values, the overall risk value is calculated using the following equation:

$$Risk = Likelihood \times Impact$$

The resulting risk values are used to prioritize vulnerability mitigation, where higher risk values indicate higher remediation priority.

3. RESULTS AND DISCUSSIONS

The security analysis of the Padak Village Administration and Letter Service Information System was conducted using the STRIDE threat modeling approach, OWASP ZAP vulnerability testing, OWASP Top 10 classification, and OWASP Risk Rating Methodology. The results provide an overview of potential threats, detected vulnerabilities, and the security risks that may affect the confidentiality, integrity, and availability of the system.

3.1 Data Flow Diagram

The Padak Village Administration and Letter Service Information System was modeled using a Level 1 Data Flow Diagram (DFD) to represent the system architecture, including processes, data flows, external entities, data stores, and trust boundaries. The Level 1 DFD provides sufficient detail to describe the interactions among system components while avoiding unnecessary abstraction, making it suitable as the basis for STRIDE threat modeling. The DFD explicitly identifies the trust boundary separating external entities from the internal application environment and illustrates the communication paths between users, the Google OAuth authentication service, the web application, and the database. Security-relevant interactions, including the Google OAuth authentication flow and HTTP/HTTPS communication paths where applicable, are represented to facilitate the identification of potential security threats associated with each DFD element. This architectural representation serves as the foundation for systematically applying the STRIDE methodology in the subsequent threat modeling process.

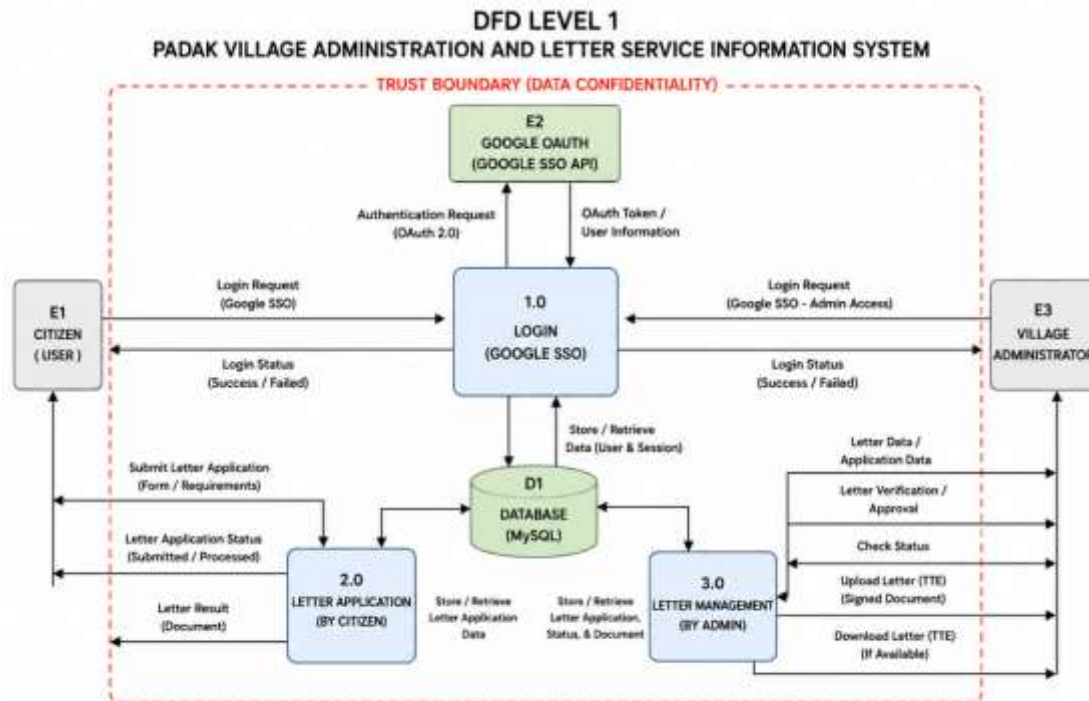


Figure 2. Level 1 Data Flow diagram

Figure 2 illustrates the Level 1 Data Flow Diagram of the Padak Village Administration and Letter Service Information System. The diagram consists of three main processes: Google SSO Login, Letter Submission, and Letter Management. Citizens authenticate through Google OAuth before accessing the application to submit administrative requests, while administrators verify, process, and approve submitted applications. User information, application records, and generated administrative documents are stored in the database. The trust boundary separates external entities from the internal application environment, enabling potential security threats to be systematically identified using the STRIDE threat modeling methodology.

3.2 STRIDE Threat Identification Results

Based on the DFD Level 1, the STRIDE threat modeling approach was applied to all DFD elements, including external entities, processes, data flows, and the data store. The identified threats provide the basis for subsequent vulnerability validation using OWASP ZAP.

Table 2. Application of STRIDE to DFD Elements

DFD Element	Elements Type	Relevant STRIDE	Threat Scenario
Citizen/User	External Entity	Spoofing	An attacker impersonates a legitimate citizen by using a

DFD Element	Elements Type	Relevant STRIDE	Threat Scenario
			compromised Google account to submit administrative requests.
Citizen/User	External Entity	Repudiation	A citizen denies submitting a letter request because the system lacks sufficient audit logs.
Village Administrator	External Entity	Spoofing	An attacker gains access to an administrator account and performs unauthorized administrative actions.
Village Administrator	External Entity	Repudiation	An administrator denies modifying the status of a submitted letter because no reliable audit trail is available.
Google SSO Login	Process	Spoofing	An attacker bypasses or abuses the authentication mechanism to impersonate a legitimate user.
Google SSO Login	Process	Tampering	Authentication requests or OAuth tokens are modified during transmission.
Google SSO Login	Process	Information Disclosure	Authentication credentials or OAuth tokens are exposed through insecure communication.
Google SSO Login	Process	Denial of Service	Excessive authentication requests prevent legitimate users from accessing the application.
Letter Submission	Process	Tampering	Malicious input, such as SQL Injection or forged requests, modifies application data during submission.

DFD Element	Elements Type	Relevant STRIDE	Threat Scenario
Letter Verification	Process	Tampering	Letter approval status is modified without authorization.
Letter Verification	Process	Repudiation	Administrative actions cannot be verified because audit logging is insufficient.
Letter Verification	Process	Elevation of Privilege	A user obtains administrative privileges and approves or rejects letter requests without authorization.
Authentication Data flow	Data Flow	Information Disclosure	Authentication information exchanged between Google OAuth and the application is intercepted during transmission.
Letter Submission Data Flow	Data Flow	Denial of Service	Excessive submission requests exhaust application resources and disrupt normal services.
User and Letter Database	Data Store	Tampering	SQL Injection modifies or deletes records stored in the database.
User and Letter Database	Data Store	Information Disclosure	Sensitive citizen information is disclosed through unauthorized database access.
User and Letter Database	Data Store	Denial of Service	Database services become unavailable because of resource exhaustion or malicious requests.

Table 2 summarizes the STRIDE threat identification for each DFD element according to its function within the Padak Village Administration and Letter Service Information System. The identified threats vary across DFD elements based on their respective security responsibilities. External entities (Citizen/User and Village Administrator) are primarily associated with Spoofing and Repudiation threats because they involve user identity and accountability. The Google SSO Login process is exposed to Spoofing, Tampering,

Information Disclosure, and Denial of Service threats due to its role in user authentication and communication with the external Google OAuth service. The Letter Submission process is primarily vulnerable to Tampering because it processes user-supplied input, whereas the Letter Verification process is additionally exposed to Repudiation and Elevation of Privilege because it involves administrative authorization and approval activities. The Authentication Data Flow is associated with Information Disclosure due to the transmission of authentication information, while the Letter Submission Data Flow is associated with Denial of Service because excessive requests may disrupt application availability. The User and Letter Database is exposed to Tampering, Information Disclosure, and Denial of Service because it stores sensitive citizen information and administrative records. These identified STRIDE threats establish the architectural threat model that is subsequently correlated with the OWASP ZAP vulnerability assessment to determine which threats are supported by observable technical evidence before being classified using OWASP Top 10 (2021) and evaluated using the OWASP Risk Rating Methodology.

3.3 OWASP ZAP Security Testing Results

OWASP ZAP was used to perform passive and active security scanning on the Padak Village Administration and Letter Service Information System. The assessment identified multiple vulnerabilities with Medium, Low, and Informational risk levels, indicating the presence of security weaknesses across several application components. These findings were subsequently classified according to the OWASP Top 10 (2021) framework for further analysis.

Figure 3. OWASP ZAP Vulnerability Summary



- > Cookie with SameSite Attribute None
- > Cookie without SameSite Attribute (3)
- > Cross-Domain JavaScript Source File Inclusion
- > HTTPS Content Available via HTTP (Systemic)
- > Strict-Transport-Security Header Not Set (Systemic)
- > X-Content-Type-Options Header Missing (Systemic)
- > Information Disclosure - Sensitive Information in URL (3)
- > Loosely Scoped Cookie
- > Modern Web Application (3)
- > Re-examine Cache-control Directives
- > Retrieved from Cache (4)
- > Session Management Response Identified (2)
- > User Agent Fuzzer (Systemic)

3.4 Vulnerability Classification Using OWASP TOP 10

The vulnerabilities identified through OWASP ZAP were classified according to the OWASP Top 10 (2021) to determine their corresponding security risk categories. This classification provides a standardized interpretation of the identified vulnerabilities and supports subsequent risk assessment. The mapping results are presented in Table 3.

Table 3. Mapping of Selected OWASP ZAP Finding to OWASP Top 10

Security Finding	Risk Level	Instances	OWASP Top 10 Category	Description
SQL Injection	High	3	A03:2021 Injection	The application is vulnerable to SQL Injection, allowing attackers to manipulate database queries and potentially access, modify, or delete sensitive data.
Absence of Anti-CSRF Tokens	Medium	4	A01:2021 Broken Access Control	Forms do not implement anti-CSRF tokens, allowing attackers to perform unauthorized requests on behalf of

Security Finding	Risk Level	Instances	OWASP Top 10 Category	Description
				authenticated users.
CSP: Failure to Define Directive with No Fallback	Medium	Systemic	A05:2021 Security Misconfiguration	Some CSP directives are missing fallback values, reducing the effectiveness of browser-enforced security policies.
CSP: Wildcard Directive	Medium	Systemic	A05:2021 Security Misconfiguration	Wildcard (*) directives allow resources to be loaded from multiple origins, increasing the attack surface.
CSP: script-src Unsafe-eval	Medium	2	A05:2021 Security Misconfiguration	The use of eval() and similar dynamic JavaScript execution functions is allowed, increasing the attack surface for code injection.
CSP: script-src Unsafe-inline	Medium	4	A05:2021 Security Misconfiguration	Inline JavaScript execution is permitted, making the application more susceptible to Cross-Site Scripting (XSS).

Security Finding	Risk Level	Instances	OWASP Top 10 Category	Description
CSP: style-src Unsafe-inline	Medium	Systemic	A05:2021 Security Misconfiguration	Inline CSS is permitted, weakening the overall Content Security Policy.
Content Security Policy (CSP) Header Not Set	Medium	Systemic	A05:2021 Security Misconfiguration	The application does not define a Content Security Policy, increasing the risk of Cross-Site Scripting (XSS) and malicious content injection.
Cross-Domain Misconfiguration	Medium	1	A05:2021 Security Misconfiguration	Cross-domain access is not properly restricted, potentially allowing unauthorized interactions between domains.
Directory Browsing	Medium	Systemic	A05:2021 Security Misconfiguration	Directory listing is enabled, exposing the application's internal file structure to attackers.
Missing Anti-clickjacking Header	Medium	Systemic	A05:2021 Security Misconfiguration	The application does not implement clickjacking protection

Security Finding	Risk Level	Instances	OWASP Top 10 Category	Description
				headers such as X-Frame-Options or frame-ancestors.
Sub Resource Integrity Attribute Missing	Medium	Systemic	A08:2021 Software and Data Integrity Failures	External CDN resources lack integrity verification, allowing compromised third-party resources to be executed.
CSP: Notices	Low	5	A05:2021 Security Misconfiguration	Informational issues related to the Content Security Policy configuration, including deprecated directives.
Cookie No HttpOnly Flag	Low	2	A07:2021 Identification and Authentication Failures	Session cookies are accessible through client-side scripts, increasing the impact of XSS attacks.
Cookie without Secure Flag	Low	1	A02:2021 Cryptographic Failures	Cookies may be transmitted over unencrypted HTTP connections, exposing sensitive information.

Security Finding	Risk Level	Instances	OWASP Top 10 Category	Description
Cookie with SameSite Attribute None	Low	1	A07:2021 Identification and Authentication Failures	Cookies configured with SameSite=None may be included in cross-site requests, increasing CSRF exposure.
Cookie without SameSite Attribute	Low	3	A07:2021 Identification and Authentication Failures	Cookies do not define the SameSite attribute, allowing them to be sent during cross-site requests.
Cross-Domain JavaScript Source File Inclusion	Low	1	A08:2021 Software and Data Integrity Failures	The application loads JavaScript resources from third-party domains, introducing dependency-related security risks.
HTTPS Content Available via HTTP	Low	Systemic	A02:2021 Cryptographic Failures	Resources accessible over HTTPS are also available through HTTP, allowing unencrypted communication.
Strict-Transport-Security Header Not Set	Low	Systemic	A05:2021 Security Misconfiguration	The HSTS header is missing, allowing

Security Finding	Risk Level	Instances	OWASP Top 10 Category	Description
				browsers to connect over insecure HTTP instead of enforcing HTTPS.
X-Content-Type-Options Header Missing	Low	Systemic	A05:2021 Security Misconfiguration	The X-Content-Type-Options: nosniff header is absent, allowing browsers to perform MIME type sniffing.

Table 3 summarizes the security vulnerabilities identified during the OWASP ZAP assessment of the Village Administration Information System. A total of 21 security findings were identified, consisting of one High-risk, eleven Medium-risk, and nine Low-risk vulnerabilities. The High-risk finding is SQL Injection, indicating that the application remains vulnerable to database manipulation attacks. Most Medium-risk findings are associated with security misconfiguration, particularly weaknesses in the implementation of Content Security Policy (CSP), missing security headers, directory browsing, and the absence of anti-CSRF protection. Meanwhile, the Low-risk findings primarily involve incomplete HTTP security header configurations, insecure cookie attributes, and weaknesses related to transport layer protection. Although these issues are categorized as Low risk, they may increase the overall attack surface when combined with other vulnerabilities. Therefore, each identified finding was further evaluated using the OWASP Risk Rating Methodology to determine its likelihood, impact, and remediation priority before security improvement recommendations were formulated.

3.5 OWASP Risk Rating Result

The identified vulnerabilities were evaluated using the OWASP Risk Rating Methodology to determine the likelihood of successful exploitation before calculating the overall risk level. In accordance with the OWASP Risk Rating framework, the likelihood assessment consists of two stages: Threat Agent Factors and Vulnerability Factors. Each sub-factor was assigned a score based on the OWASP Risk Rating criteria, and the intermediate scores were calculated by averaging the corresponding sub-factors. Presenting both the raw scores and the intermediate averages provides a transparent and auditable calculation trail, allowing each vulnerability to be individually prioritized before the final risk assessment.

Table 4. Threat Agent Factors

Vulnerability	<i>Skill Level</i>	<i>Motive</i>	<i>Opportunity</i>	<i>Size</i>	Threat Agent Score
<i>SQL Injection</i>	9	9	9	9	9
<i>Absence of Anti-CSRF Tokens</i>	6	9	7	9	7.75
<i>CSP: Failure to Define Directive with No Fallback</i>	3	4	7	9	5.75
<i>CSP: Wildcard Directive</i>	3	4	7	9	5.75
<i>CSP: script-src Unsafe-eval</i>	6	9	7	9	7.75
<i>CSP: script-src Unsafe-inline</i>	6	9	7	9	7.75
<i>CSP: style-src Unsafe-inline</i>	3	4	7	9	5.75
<i>Content Security Policy (CSP) Header Not Set</i>	3	4	7	9	5.75
<i>Cross-Domain Misconfiguration</i>	6	9	7	9	7.75
<i>Directory Browsing</i>	3	4	9	9	6.25
<i>Missing Anti-clickjacking Header</i>	3	4	7	9	5.75
<i>Sub Resource Integrity</i>	6	9	7	9	7.75

Vulnerability	<i>Skill Level</i>	<i>Motive</i>	<i>Opportunity</i>	<i>Size</i>	Threat Agent Score
<i>Attribute Missing</i>					
<i>CSP: Notices</i>	1	1	7	9	4.5
<i>Cookie No HttpOnly Flag</i>	3	4	7	9	5.75
<i>Cookie without Secure Flag</i>	3	4	7	9	5.75
<i>Cookie with SameSite Attribute None</i>	3	4	7	9	5.75
<i>Cookie without SameSite Attribute</i>	3	4	7	9	5.75
<i>Cross-Domain JavaScript Source File Inclusion</i>	6	9	7	9	7.75
<i>HTTPS Content Available via HTTP</i>	3	4	9	9	6.25
<i>Strict-Transport-Security Header Not Set</i>	3	4	7	9	5.75
<i>X-Content-Type-Options Header Missing</i>	3	4	7	9	5.75

Table 4 presents the Threat Agent assessment for all 21 identified vulnerabilities. The values assigned to Skill Level, Motive, Opportunity, and Size follow the OWASP Risk Rating scoring criteria described in the methodology. Rather than presenting only the raw factor values, the Threat Agent Score is also reported as the arithmetic mean of the four sub-factors:

$$\text{Threat Agent} = \frac{(\text{Skill Level} + \text{Motive} + \text{Opportunity} + \text{Size})}{4}$$

This intermediate score provides an auditable calculation trail showing how the individual factor scores contribute to the likelihood assessment. For example, SQL Injection obtained the highest Threat Agent Score (9), indicating that exploitation requires highly capable attackers with strong motivation and broad opportunities. In contrast, CSP: Notices obtained the lowest Threat Agent Score (4.5), reflecting a comparatively lower threat capability and motivation. These intermediate scores are subsequently combined with the Vulnerability Scores to calculate the final Likelihood value for each vulnerability.

Table 5. Vulnerability Factors

Vulnerability	Ease of Discovery	Ease of Exploit	Awareness	Intrusion Detection	Vulnerability Score
<i>SQL Injection</i>	9	9	9	9	8.75
<i>Absence of Anti-CSRF Tokens</i>	6	9	7	9	7.75
<i>CSP: Failure to Define Directive with No Fallback</i>	3	4	7	9	7.75
<i>CSP: Wildcard Directive</i>	3	4	7	9	7.75
<i>CSP: script-src Unsafe-eval</i>	6	9	7	9	7.75
<i>CSP: script-src Unsafe-inline</i>	6	9	7	9	7.75
<i>CSP: style-src Unsafe-inline</i>	3	4	7	9	7.75
<i>Content Security Policy (CSP) Header Not Set</i>	3	4	7	9	7.75
<i>Cross-Domain Misconfiguration</i>	6	9	7	9	6.5

Vulnerability	Ease of Discovery	Ease of Exploit	Awareness	Intrusion Detection	Vulnerability Score
<i>Directory Browsing</i>	3	4	9	9	9
<i>Missing Anti-clickjacking Header</i>	3	4	7	9	8
<i>Sub Resource Integrity Attribute Missing</i>	6	9	7	9	6.5
<i>CSP: Notices</i>	1	1	7	9	7.25
<i>Cookie No HttpOnly Flag</i>	3	4	7	9	7.75
<i>Cookie without Secure Flag</i>	3	4	7	9	7.75
<i>Cookie with SameSite Attribute None</i>	3	4	7	9	7.75
<i>Cookie without SameSite Attribute</i>	3	4	7	9	7.75
<i>Cross-Domain JavaScript Source File Inclusion</i>	6	9	7	9	6.5
<i>HTTPS Content Available via HTTP</i>	3	4	9	9	7.75
<i>Strict-Transport-Security Header Not Set</i>	3	4	7	9	7.75

Vulnerability	Ease of Discovery	Ease of Exploit	Awareness	Intrusion Detection	Vulnerability Score
<i>X-Content-Type-Options Header Missing</i>	3	4	7	9	7.75

Table 5 summarizes the Vulnerability assessment for the 21 identified vulnerabilities. The scores assigned to Ease of Discovery, Ease of Exploit, Awareness, and Intrusion Detection were determined according to the OWASP Risk Rating scoring guidelines. To improve transparency and reproducibility, the Vulnerability Score is presented as the arithmetic mean of the four sub-factors:

$$Vulnerability = \frac{(Ease\ of\ Discovery + Ease\ of\ Exploit + Awareness + Intrusion\ Detection)}{4}$$

Including this intermediate score establishes a clear calculation trail from the individual OWASP factor ratings to the overall likelihood assessment. Among the identified vulnerabilities, Directory Browsing obtained the highest Vulnerability Score (9), indicating that it is highly discoverable and relatively easy to exploit. Conversely, Cross-Domain Misconfiguration, Sub Resource Integrity Attribute Missing, and Cross-Domain JavaScript Source File Inclusion obtained lower scores (6.5), reflecting comparatively lower exploitability. These Vulnerability Scores are then combined with the corresponding Threat Agent Scores to calculate the Likelihood for each vulnerability, which serves as the basis for subsequent risk prioritization.

3.5.1 Likelihood Assessment

After calculating the Threat Agent Score and Vulnerability Score, the Likelihood of each vulnerability was determined following the OWASP Risk Rating Methodology. The Likelihood score represents the probability that a vulnerability can be successfully exploited by considering both the attacker's capability and the technical characteristics of the vulnerability. It was calculated using the following equation:

$$Likelihood = \frac{Threat\ Agent + Vulnerability}{2}$$

The intermediate Threat Agent and Vulnerability scores presented in Tables 3 and 4 provide a transparent calculation trail for deriving the Likelihood score of each vulnerability. Table 6 summarizes the resulting Likelihood values, which were subsequently used together with the Technical Impact and Business Impact scores to determine the Overall Risk level.

Table 6. Likelihood Assessment

Vulnerability	Threat Agent Factors	Vulnerability Factors	Likelihood
<i>SQL Injection</i>	9	8.75	8.875
<i>Absence of Anti-CSRF Tokens</i>	7.75	7.75	7.75
<i>CSP: Failure to Define Directive with No Fallback</i>	5.75	7.75	6.75
<i>CSP: Wildcard Directive</i>	5.75	7.75	6.75
<i>CSP: script-src Unsafe-eval</i>	7.75	7.75	7.75
<i>CSP: script-src Unsafe-inline</i>	7.75	7.75	7.75
<i>CSP: style-src Unsafe-inline</i>	5.75	7.75	6.75
<i>Content Security Policy (CSP) Header Not Set</i>	5.75	7.75	6.75
<i>Cross-Domain Misconfiguration</i>	7.75	6.5	7.125
<i>Directory Browsing</i>	6.25	9	7.625
<i>Missing Anti-clickjacking Header</i>	5.75	8	6.875
<i>Sub Resource Integrity Attribute Missing</i>	7.75	6.5	7.125
<i>CSP: Notices</i>	4.5	7.25	5.875
<i>Cookie No HttpOnly Flag</i>	5.75	7.75	6.75
<i>Cookie without Secure Flag</i>	5.75	7.75	6.75
<i>Cookie with SameSite Attribute None</i>	5.75	7.75	6.75
<i>Cookie without SameSite Attribute</i>	5.75	7.75	6.75
<i>Cross-Domain JavaScript Source File Inclusion</i>	7.75	6.5	7.125

Vulnerability	<i>Threat Agent Factors</i>	<i>Vulnerability Factors</i>	<i>Likelihood</i>
<i>HTTPS Content Available via HTTP</i>	6.25	7.75	7
<i>Strict-Transport-Security Header Not Set</i>	5.75	7.75	6.75
<i>X-Content-Type-Options Header Missing</i>	5.75	7.75	6.75

Table 6 presents the Likelihood scores calculated for the 21 identified vulnerabilities. The values were obtained by averaging the Threat Agent Score and the Vulnerability Score, providing an auditable and reproducible calculation process for each vulnerability. The results indicate that SQL Injection achieved the highest Likelihood score (8.875), suggesting the greatest probability of successful exploitation due to the combination of highly capable threat agents and a technically exploitable vulnerability. Other vulnerabilities, including Absence of Anti-CSRF Tokens, CSP: script-src unsafe-eval, and CSP: script-src unsafe-inline, also exhibited relatively high Likelihood scores (7.75), indicating substantial exploitation potential. Conversely, CSP: Notices produced the lowest Likelihood score (5.875), reflecting a comparatively lower probability of successful exploitation. These Likelihood scores were subsequently combined with the Technical Impact and Business Impact assessments to determine the Overall Risk and establish remediation priorities.

Tabel 7. Technical Impact Factors

Vulnerability	Loss of Confidentialit y	Loss of Integrit y	Lost of Availabilit y	Lost of Accountabilit y	Technica l Impact Score
<i>SQL Injection</i>	9	9	7	9	8.5
<i>Absence of Anti- CSRF Tokens</i>	2	7	1	7	4.25
<i>CSP: Failure to Define Directive with No Fallback</i>	2	3	1	1	1.75
<i>CSP: Wildcard Directive</i>	6	5	1	7	4.75

Vulnerability	Loss of Confidentiality	Loss of Integrity	Lost of Availability	Lost of Accountability	Technical Impact Score
<i>CSP: script-src Unsafe-eval</i>	7	7	1	7	5.5
<i>CSP: script-src Unsafe-inline</i>	7	7	1	7	5.5
<i>CSP: style-src Unsafe-inline</i>	2	3	1	1	1.75
<i>Content Security Policy (CSP) Header Not Set</i>	6	5	1	7	4.75
<i>Cross-Domain Misconfiguration</i>	6	3	1	1	2.75
<i>Directory Browsing</i>	6	1	1	1	2.25
<i>Missing Anti-clickjacking Header</i>	2	5	1	7	3.75
<i>Sub Resource Integrity Attribute Missing</i>	7	7	5	7	6.5
<i>CSP: Notices</i>	2	2	1	1	1.5
<i>Cookie No HttpOnly Flag</i>	6	1	1	7	3.75
<i>Cookie without Secure Flag</i>	6	3	1	7	4.25
<i>Cookie with SameSite Attribute None</i>	2	5	1	7	3.75

Vulnerability	Loss of Confidentiality	Loss of Integrity	Loss of Availability	Loss of Accountability	Technical Impact Score
<i>Cookie without SameSite Attribute</i>	2	5	1	7	3.75
<i>Cross-Domain JavaScript Source File Inclusion</i>	7	7	5	7	6.5
<i>HTTPS Content Available via HTTP</i>	6	3	1	7	4.25
<i>Strict-Transport-Security Header Not Set</i>	6	3	1	7	4.25
<i>X-Content-Type-Options Header Missing</i>	6	5	1	7	4.75

Table 7 presents the Technical Impact assessment used to estimate the consequences of successful exploitation on the system. Scores were assigned according to the OWASP Risk Rating Methodology by evaluating four security objectives: confidentiality, integrity, availability, and accountability. For example, SQL Injection received the highest scores because successful exploitation could expose sensitive information, modify database contents, disrupt application services, and reduce attacker traceability. In contrast, configuration-related findings such as CSP: Notices received lower scores because their direct technical consequences are comparatively limited. These sub-factor scores provide a transparent basis for calculating the Technical Impact value for each vulnerability.

Tabel 8. Business Impact

Vulnerability	Financial Damage	Reputation Damage	Non-compliance	Privacy Violation	Business Impact Score
<i>SQL Injection</i>	7	9	7	7	7.5

Vulnerability	<i>Financial Damage</i>	<i>Reputation Damage</i>	<i>Non- compliance</i>	<i>Privacy Violation</i>	Business Impact Score
<i>Absence of Anti-CSRF Tokens</i>	3	5	5	5	4.5
<i>CSP: Failure to Define Directive with No Fallback</i>	1	1	2	3	1.75
<i>CSP: Wildcard Directive</i>	3	4	2	3	3
<i>CSP: script-src Unsafe-eval</i>	3	5	5	5	4.5
<i>CSP: script-src Unsafe-inline</i>	1	1	2	3	1.75
<i>CSP: style-src Unsafe-inline</i>	3	4	2	1	2.5
<i>Content Security Policy (CSP) Header Not Set</i>	3	5	5	5	4.5
<i>Cross-Domain Misconfiguration</i>	3	5	5	5	4.5
<i>Directory Browsing</i>	1	1	2	3	1.75
<i>Missing Anti-clickjacking Header</i>	3	5	5	5	4.5
<i>Sub Resource Integrity Attribute Missing</i>	3	4	2	3	3
<i>CSP: Notices</i>	3	4	2	5	3.5

Vulnerability	<i>Financial Damage</i>	<i>Reputation Damage</i>	<i>Non- compliance</i>	<i>Privacy Violation</i>	Business Impact Score
<i>Cookie No HttpOnly Flag</i>	1	4	2	3	2.5
<i>Cookie without Secure Flag</i>	3	5	5	5	4.5
<i>Cookie with SameSite Attribute None</i>	3	4	2	5	3.5
<i>Cookie without SameSite Attribute</i>	3	4	2	5	3.5
<i>Cross-Domain JavaScript Source File Inclusion</i>	7	5	5	5	5.5
<i>HTTPS Content Available via HTTP</i>	3	5	5	5	4.5
<i>Strict-Transport- Security Header Not Set</i>	3	5	5	5	4.5
<i>X-Content-Type- Options Header Missing</i>	3	4	2	3	3

Table 8 summarizes the Business Impact assessment. Scores were assigned based on the expected organizational consequences of each vulnerability, considering financial loss, reputational damage, regulatory non-compliance, and privacy violations. Vulnerabilities capable of exposing or manipulating sensitive organizational data, such as SQL Injection, received higher business impact scores because they may lead to financial losses, reputational damage, regulatory issues, and privacy breaches. Conversely, configuration weaknesses with limited direct organizational consequences received lower scores. This assessment provides a transparent justification for the Business Impact values used in the subsequent risk calculation.

Table 9. Impact Calculation

Vulnerability	<i>Technical Impact</i>	<i>Business Impact</i>	<i>Impact</i>
<i>SQL Injection</i>	8.5	7.5	8
<i>Absence of Anti-CSRF Tokens</i>	4.25	4.5	4.375
<i>CSP: Failure to Define Directive with No Fallback</i>	1.75	1.75	175
<i>CSP: Wildcard Directive</i>	4.75	3	3.875
<i>CSP: script-src Unsafe-eval</i>	5.5	4.5	5
<i>CSP: script-src Unsafe-inline</i>	5.5	1.75	3.625
<i>CSP: style-src Unsafe-inline</i>	1.75	2.5	2.125
<i>Content Security Policy (CSP) Header Not Set</i>	4.75	4.5	4.625
<i>Cross-Domain Misconfiguration</i>	2.75	4.5	3.625
<i>Directory Browsing</i>	2.25	1.75	2
<i>Missing Anti-clickjacking Header</i>	3.75	4.5	4.125
<i>Sub Resource Integrity Attribute Missing</i>	6.5	3	4.75
<i>CSP: Notices</i>	1.5	3.5	2.5
<i>Cookie No HttpOnly Flag</i>	3.5	2.5	3.125
<i>Cookie without Secure Flag</i>	4.25	4.5	4.375
<i>Cookie with SameSite Attribute None</i>	3.75	3.5	3.625
<i>Cookie without SameSite Attribute</i>	3.75	3.5	3.625
<i>Cross-Domain JavaScript Source File Inclusion</i>	6.5	5.5	6

Vulnerability	<i>Technical Impact</i>	<i>Business Impact</i>	<i>Impact</i>
<i>HTTPS Content Available via HTTP</i>	4.25	4.5	4.375
<i>Strict-Transport-Security Header Not Set</i>	4.25	4.5	4.375
<i>X-Content-Type-Options Header Missing</i>	4.75	3	3.875

The overall Impact score was calculated as the arithmetic mean of the Technical Impact and Business Impact values for each vulnerability. This calculation provides an auditable assessment of the severity of successful exploitation from both technical and organizational perspectives. As shown in Table 7, SQL Injection obtained the highest Impact score (8) due to its severe consequences for both system security and organizational operations. Cross-Domain JavaScript Source File Inclusion ranked second with an Impact score of 6, while CSP: Failure to Define Directive with No Fallback produced the lowest Impact score (1.75). These Impact values are subsequently combined with the Likelihood scores to determine the Overall Risk and prioritize mitigation efforts.

Table 10. Likelihood and Impact Level Scale

Likelihood and Impact Levels	
0 to <3	LOW
3 to <6	MEDIUM
6 to 9	HIGH

The likelihood and impact scores calculated for each identified vulnerability were classified using the OWASP Risk Rating scale shown in Table 10. Likelihood scores were categorized as Low (0–<3), Medium (3–<6), or High (6–9), and the same classification was applied to impact scores. These categorical values were then mapped to the OWASP Risk Rating Matrix to determine the overall risk level of each vulnerability individually. This approach preserves the relative severity of every security finding and provides a transparent basis for prioritizing mitigation activities. The OWASP Risk Rating Matrix used in this study is presented in Table X, while the resulting overall risk classification for all identified vulnerabilities is summarized in Table 11.

Table 11. Overall Risk Severity Matrix

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

Using the OWASP Risk Rating Matrix, each vulnerability was assigned an overall risk level based on the combination of its likelihood and impact categories. The resulting risk levels were classified into five categories: Note, Low, Medium, High, and Critical. This per-vulnerability assessment provides an auditable prioritization of security findings, ensuring that remediation efforts focus first on vulnerabilities with the greatest potential impact on the security of the application.

Table 12. Overall Risk Rating Result

Vulnerability	Likelihood	Impact	Overall Risk
SQL Injection	8.875 (High)	8 (High)	Critical
Absence of Anti-CSRF Tokens	7.75 (High)	4.375 (Medium)	High
CSP: Failure to Define Directive with No Fallback	6.75 (High)	1.75 (Low)	Medium
CSP: Wildcard Directive	6.75 (High)	3.875 (Medium)	High
CSP: script-src Unsafe-eval	7.75 (High)	5 (Medium)	High
CSP: script-src Unsafe-inline	7.75 (High)	3.625 (Medium)	High
CSP: style-src Unsafe-inline	6.75 (High)	2.125 (Low)	Medium
Content Security Policy (CSP) Header Not Set	6.75 (High)	4.625 (Medium)	High
Cross-Domain Misconfiguration	7.125 (High)	3.625 (Medium)	High
Directory Browsing	7.625 (High)	2 (Low)	Medium

Vulnerability	Likelihood	Impact	Overall Risk
Missing Anti-clickjacking Header	6.875 (High)	4.125 (Medium)	High
Sub Resource Integrity Attribute Missing	7.125 (High)	4.75 (Medium)	High
CSP: Notices	5.875 (Medium)	2.5 (Low)	Low
Cookie No HttpOnly Flag	6.75 (High)	3.125 (Medium)	High
Cookie without Secure Flag	6.75 (High)	4.375 (Medium)	High
Cookie with SameSite Attribute None	6.75 (High)	3.625 (Medium)	High
Cookie without SameSite Attribute	6.75 (High)	3.625 (Medium)	High
Cross-Domain JavaScript Source File Inclusion	7.125 (High)	6 (High)	Critical
HTTPS Content Available via HTTP	7 (High)	4.375 (Medium)	High
Strict-Transport-Security Header Not Set	6.75 (High)	4.375 (Medium)	High
X-Content-Type-Options Header Missing	6.75 (High)	3.875 (Medium)	High

The results indicate that SQL Injection and Cross-Domain JavaScript Source File Inclusion were classified as Critical risk because both vulnerabilities exhibit high likelihood and high impact scores, making them the highest remediation priority. In addition, fifteen vulnerabilities were classified as High, indicating that they are highly exploitable and could significantly affect the confidentiality, integrity, or availability of the application if left unaddressed. Three vulnerabilities were categorized as Medium, namely CSP: Failure to Define Directive with No Fallback, CSP: style-src Unsafe-inline, and Directory Browsing, while CSP: Notices was classified as Low due to its relatively limited impact despite requiring corrective action. These results establish a clear prioritization of remediation based on the individual risk rating of each identified vulnerability rather than an aggregated application-level score.

3.6 Mitigation Recommendations

Based on the security assessment results obtained from OWASP ZAP and the individual risk ratings determined using the OWASP Risk Rating Methodology, mitigation

recommendations were developed for each identified vulnerability. The recommendations were formulated by considering the root cause of each vulnerability, the corresponding OWASP security guidance, and secure coding best practices. To support risk-based remediation, each recommendation is associated with its individual risk level and implementation priority. Critical-risk vulnerabilities are assigned immediate priority because they present the highest likelihood of exploitation and the greatest potential impact on the security of the application. High-risk vulnerabilities are recommended for remediation immediately after critical issues, whereas Medium- and Low-risk vulnerabilities should be addressed during subsequent security improvement activities. Since the implementation of security controls was outside the scope of this study, the verification methods presented in this table represent recommended validation procedures to be performed after the proposed mitigations have been implemented. Table 13 presents the mitigation recommendations for all identified vulnerabilities.

Table 13. Mitigation Recommendations

Vulnerability	Risk Level	Implementation Priority	Recommendation Mitigation	Recommended Verification
SQL Injection	Critical	Immediate	Use prepared statements (parameterized queries), validate and sanitize user input, apply whitelist validation, and enforce least-privilege database accounts.	Re-scan using OWASP ZAP Active Scan and manually verify that SQL injection payloads are rejected.
Absence of Anti-CSRF Tokens	High	Immediate	Implement unique CSRF tokens for every form and validate the token on the server before processing requests.	Verify CSRF protection through manual testing and OWASP ZAP.
CSP: Failure to Define Directive with No Fallback	Medium	Short-term	Define a secure fallback directive (e.g., default-src 'self') within the Content Security Policy.	Verify the CSP header using OWASP ZAP and browser Developer Tools.

Vulnerability	Risk Level	Implementation Priority	Recommendation Mitigation	Recommended Verification
CSP: Wildcard Directive	High	Immediate	Replace wildcard (*) directives with an explicit whitelist of trusted domains.	Verify that wildcard directives are no longer present in the CSP header.
CSP: script-src Unsafe-eval	High	Immediate	Remove the unsafe-eval directive and replace dynamic JavaScript evaluation with secure alternatives.	Verify the updated CSP policy using OWASP ZAP.
CSP: script-src Unsafe-inline	High	Immediate	Move inline scripts to external JavaScript files or use CSP nonce/hash mechanisms.	Confirm that inline scripts are blocked according to the CSP policy.
CSP: style-src Unsafe-inline	Medium	Short-term	Move inline styles to external CSS files or use CSP nonce/hash values.	Verify that inline styles comply with the updated CSP configuration.
Content Security Policy (CSP) Header Not Set	High	Immediate	Configure a Content Security Policy header on the web server to restrict trusted resource sources.	Verify that the CSP header is included in every HTTP response.
Cross-Domain Misconfiguration	High	Immediate	Configure CORS to allow access only from trusted domains and avoid Access-Control-Allow-Origin: *.	Verify CORS configuration using OWASP ZAP and manual inspection.
Directory Browsing	Medium	Short-term	Disable directory listing on the web	Confirm that directory

Vulnerability	Risk Level	Implementation Priority	Recommendation Mitigation	Recommended Verification
			server and restrict direct access to sensitive directories.	indexing is disabled through manual testing
Missing Anti-clickjacking Header	High	Immediate	Configure the X-Frame-Options header (DENY or SAMEORIGIN) or use the CSP frame-ancestors directive.	Verify response headers using OWASP ZAP.
Sub Resource Integrity Attribute Missing	High	Immediate	Add Subresource Integrity (SRI) attributes to externally loaded JavaScript and CSS resources.	Verify that all external resources include valid integrity attributes
CSP: Notices	Low	Medium-term	Review and refine the CSP configuration by removing deprecated or unnecessary directives.	Verify that CSP warnings are no longer reported.
Cookie No HttpOnly Flag	High	Immediate	Set the HttpOnly attribute on all session cookies.	Verify cookie attributes using browser Developer Tools or OWASP ZAP.
Cookie without Secure Flag	High	Immediate	Configure cookies with the Secure attribute so they are transmitted only over HTTPS.	Verify the Secure attribute in HTTP responses.
Cookie with SameSite Attribute None	High	Immediate	Replace SameSite=None with SameSite=Lax	Verify cookie configuration

Vulnerability	Risk Level	Implementation Priority	Recommendation Mitigation	Recommended Verification
			or SameSite=Strict where cross-site access is unnecessary, and enable the Secure attribute.	through browser inspection.
Cookie without SameSite Attribute	High	Immediate	Add the SameSite=Lax or SameSite=Strict attribute to reduce CSRF exposure.	Verify that cookies include the SameSite attribute.
Cross-Domain JavaScript Source File Inclusion	Critical	Immediate	Load JavaScript only from trusted domains and apply Subresource Integrity (SRI) for external resources.	Verify the origin and integrity of external JavaScript resources.
HTTPS Content Available via HTTP	High	Immediate	Redirect all HTTP traffic to HTTPS and disable unsecured HTTP access.	Verify automatic HTTP-to-HTTPS redirection using OWASP ZAP.
Strict-Transport-Security Header Not Set	High	Immediate	Configure the HTTP Strict Transport Security (HSTS) header to enforce HTTPS connections.	Verify that the HSTS header is present in HTTP responses..
X-Content-Type-Options Header Missing	High	Immediate	Configure the web server to include the X-Content-Type-Options: nosniff header in all HTTP responses.	Verify the presence of the header using OWASP ZAP.

The mitigation recommendations presented in Table 13 provide a structured remediation plan based on the individual risk assessment results rather than an aggregated application-level risk score. The two Critical vulnerabilities, namely SQL Injection and Cross-Domain JavaScript Source File Inclusion, should be addressed first because they exhibit both high likelihood and high impact, making them the highest remediation priority. The remaining High-risk vulnerabilities should subsequently be mitigated to reduce the overall attack surface of the application, while Medium- and Low-risk vulnerabilities should be resolved as part of continuous security hardening. After implementing the proposed security controls, a follow-up security assessment using OWASP ZAP together with complementary manual testing should be conducted to verify that the identified vulnerabilities have been successfully mitigated and to evaluate the remaining residual risk.

4. CONCLUSION

This study evaluated the security of the Padak Village Administration and Letter Service Information System by applying STRIDE for threat modeling, OWASP ZAP for automated vulnerability assessment, OWASP Top 10 (2021) for vulnerability classification, and the OWASP Risk Rating Methodology for evaluating the likelihood and impact of each confirmed vulnerability. A total of 21 validated vulnerabilities were included in the risk assessment after excluding seven informational findings that did not represent directly exploitable security weaknesses. The assessment results consisted of 2 Critical-, 15 High-, 3 Medium-, and 1 Low-risk vulnerabilities. These findings provide a basis for prioritizing remediation according to the calculated risk level of each confirmed vulnerability. This study has several limitations. First, the vulnerability assessment relied primarily on the automated scanning capabilities of OWASP ZAP, which may not detect all security weaknesses and may produce false positive or false negative findings. Second, the evaluation covered only the authentication roles available in the system (Administrator and User), so vulnerabilities associated with other privilege levels or deployment configurations were not assessed. Third, the study did not include source-code review or manual secure code analysis, meaning that implementation-level weaknesses not observable through runtime testing may remain undiscovered. Finally, the proposed mitigation measures were not implemented within the scope of this research, and no post-remediation security testing was performed. Future work should combine automated scanning with manual penetration testing and source-code review, evaluate additional authentication scenarios and user roles, implement the recommended security controls, and conduct follow-up security assessments to verify the effectiveness of the applied mitigations and assess the remaining residual risk.

5. REFERENCES

- Das, Popy, Md. Rashid Al Asif, Sohely Jahan, Kawsar Ahmed, Francis M. Bui, and Rahamatullah Khondoker. 2024. "STRIDE-Based Cybersecurity Threat Modeling , Risk Assessment and Treatment of an In-Vehicle Infotainment System." *Vehicles* 6:1140–63.
- Foundation, OWASP. 2021. "OWASP Top 10: The Ten Most Critical Web Application Security Risks." Retrieved July 30, 2026 (<https://owasp.org/www-project-top-ten/>).

- Microsoft. 2024. "Threat Modeling Tool Threats." Retrieved July 30, 2026 (<https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats>).
- Mukrodin, Rohmat Taufiq, and Deskal Dwi Rayananda. 2023. "Implementasi Single Sign On Dengan OAuth 2.0 Untuk Efektifitas Login." *Jurnal Teknik* 12(02):36–50.
- Rifat, Md. Asif Khan, Yeasmin Sultana, and B. M. Mainul Hossain. 2023. "Vulnerabilities Assessment of Financial and Government Websites : A Developing Country Perspective." *International Journal of Information Engineering and Electronic Business (IJIEEB)* 15(5):42–53. doi: 10.5815/ijieeb.2023.05.05.
- Salmuasih, and Mukhammad Andri Setiawan. 2023. "Evaluasi Penerapan Single Sign-On SAML Dan OAuth 2.0: Studi Pada Perguruan Tinggi Yogyakarta." *Jurnal Sistem Informasi* 10(1):41–49. doi: 10.30656/jsii.v10i1.6186.
- Wisnu, Muhammad, and Benfano Soewito. 2026. "Security Assessment Based on OWASP Top 10 Using SonarQube and ZAP on Export and Import Applications in the LNSW." *INTENSIF: Jurnal Ilmiah Penelitian Dan Penerapan Teknologi Sistem Informasi* 10(1):36–53.